



Sensores



**Sergio
Velásquez**

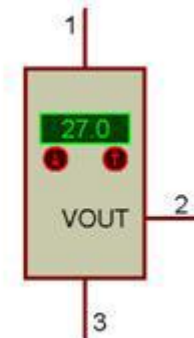
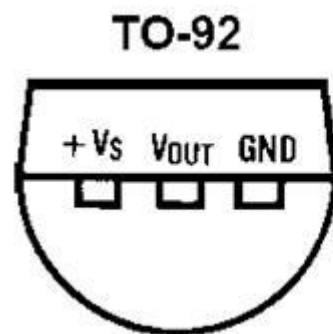
- La implementación de sensores en sistemas microcontrolados es de gran utilidad en muchas de sus aplicaciones, estos permiten obtener lecturas de variables como temperatura, aceleración, presión, humedad, velocidad, luminosidad, contraste, entre otras. Su aplicación es importante en sistemas de control, robótica, e instrumentación.

Sensor de temperatura LM35

- El sensor LM35, es un dispositivo activo de 3 terminales que permite adquirir la temperatura ambiente en rangos de -55 a 150 grados Celsius o centígrados. Este dispositivo es de fácil implementación dado que solo cuenta con dos terminales de polarización, y una salida de voltaje directamente proporcional a la temperatura. Este sensor puede ser polarizado de 4 a 30 voltios y tiene una salida de 10m voltios por cada grado Celsius. La apariencia física del sensor y su distribución de pines, así como la vista en ISIS, son las que se pueden ver en las siguientes figuras:



Sergio
Velásquez



Sensor de temperatura LM35

Para realizar la lectura del voltaje de salida del sensor se implementa en el microcontrolador el módulo ADC. La máxima salida del sensor es 1,5 voltios, cuando la temperatura es 150 grados Celsius. Por esto es importante cambiar el valor de referencia positiva del convertidor análogo digital, con el fin de mejorar la resolución de la medida de voltaje. Para el ejemplo de este capítulo se configurará el voltaje de referencia positivo del ADC, en 2,5 voltios. Cambiando la referencia positiva a 2,5 voltios el convertidor entregará un resultado binario de 1023 cuando el voltaje a convertir es de 2,5 voltios. Para el caso de este sensor, se verá definido por las siguientes relaciones:

$$\frac{1023}{2,5V} = \frac{R_{adc}}{V_{adc}}$$

Donde R_{adc} es el resultado binario de la conversión AD. De esta ecuación se puede deducir que el voltaje V_{adc} , leído por el convertidor AD, es:

$$V_{adc} = \frac{(2,5V)(R_{adc})}{1023}$$

Sensor de temperatura LM35

Trabajando con la relación del sensor que es: 10m voltios por cada grado Celsius, se puede plantear la siguiente ecuación: $\frac{10mV}{1^{\circ}C} = \frac{V_{adc}}{n^{\circ}C}$

Donde n es la temperatura en grados Celsius, que está registrando el sensor, de esta ecuación se puede deducir que la temperatura n es:

$$n^{\circ}C = \frac{(1^{\circ}C)(V_{adc})}{10mV}$$

Remplazando la ecuación V_{adc} , en $n^{\circ}C$, se obtiene la siguiente relación:

$$n^{\circ}C = \frac{(2,5)(R_{adc})}{10,23} = 0,244R_{adc}$$

Esta relación debe ser implementada en la conversión AD, en el programa del PIC.

Sensor de temperatura LM35

Para contextualizar el uso de este sensor se puede observar y analizar el siguiente código fuente para un PIC 16F877A:

```
//Definición de pines del LCD  
sbit LCD_RS at RB4_bit;  
sbit LCD_EN at RB5_bit;  
sbit LCD_D7 at RB3_bit;  
sbit LCD_D6 at RB2_bit;  
sbit LCD_D5 at RB1_bit;  
sbit LCD_D4 at RB0_bit;  
//Definición de los TRIS del LCD  
sbit LCD_RS_Direction at TRISB4_bit;  
sbit LCD_EN_Direction at TRISB5_bit;  
sbit LCD_D7_Direction at TRISB3_bit;  
sbit LCD_D6_Direction at TRISB2_bit;  
sbit LCD_D5_Direction at TRISB1_bit;  
sbit LCD_D4_Direction at TRISB0_bit;
```

Sensor de temperatura LM35

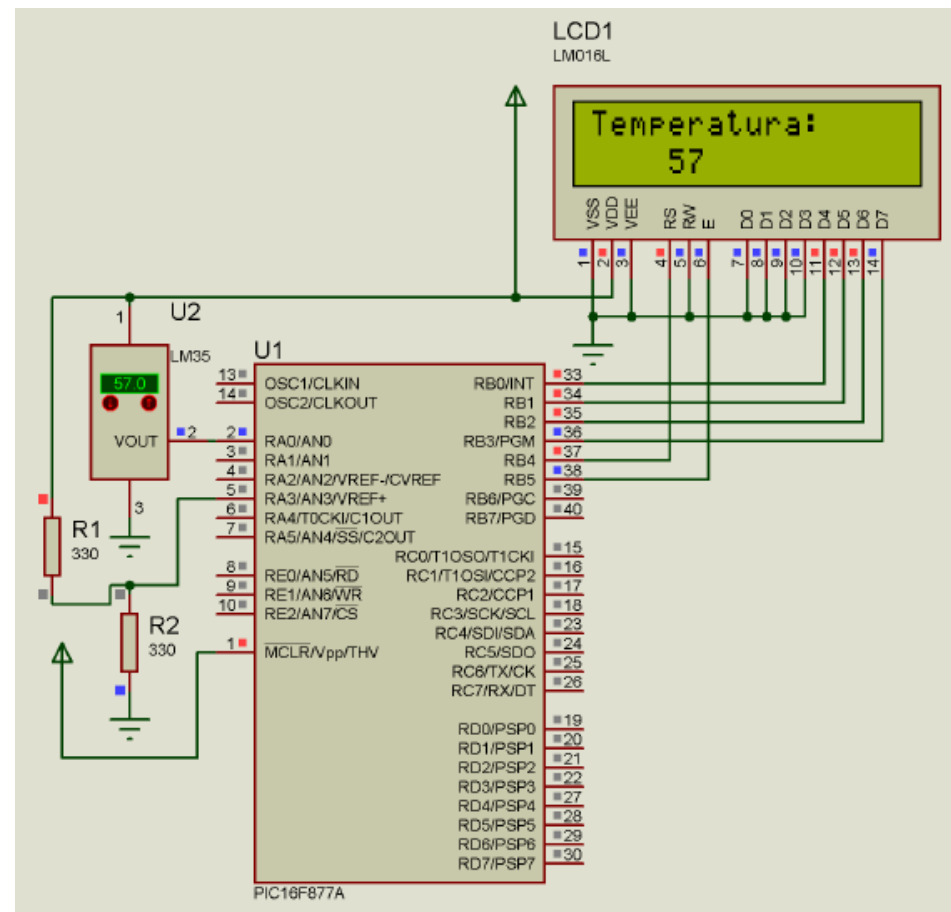
```
void main( void )
{
    //Declaración de variables.
    unsigned int Radc, Teml;
    float Tem;
    char Text[16];
    //Configura el módulo ADC con el pin AN3
    //como voltaje de referencia positiva.
    ADCON1 = 0b11000001;
    //Inicio del LCD.
    Lcd_Init();
    //Borrado del cursor.
    Lcd_Cmd(_LCD_CURSOR_OFF);
    //Impresión de texto.
    Lcd_Out( 1, 1, "Temperatura:");
```

```
while(1) //Bucle infinito.
{
    //Lectura del canal 0 del ADC.
    Radc = ADC_Read(0);
    //Uso de la ecuación (13.5).
    Tem = 0.244*Radc;
    //Se convierte el resultado a un número entero.
    Teml = Tem;
    //Se convierte el número entero a una cadena
    de caracteres.
    IntToStr(Teml,Text );
    //Se imprime el resultado.
    Lcd_Out( 2, 1,Text);
    //Retardo de 100m segundos.
    delay_ms(100);
}
}
```

Sensor de temperatura LM35

Terminada la edición y compilación del programa se debe construir un circuito en ISIS con los siguientes dispositivos: 16F877A, RES, LM35, y LM016L, este se puede apreciar en la siguiente figura:

El arreglo de resistencias de 330Ω permite hacer un divisor de voltaje para crear la referencia de 2,5 voltios. Durante la marcha de la simulación se puede cambiar el valor de la temperatura en el sensor LM35, para ver su funcionamiento

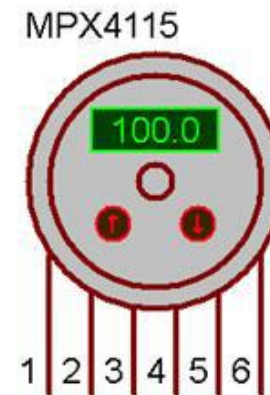
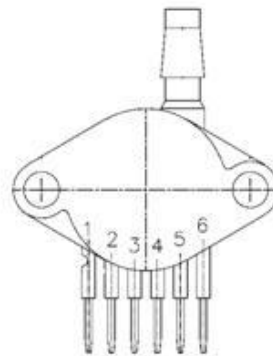


Sensores de presión

- La presión es una variable física, representada por un vector que está definido como la fuerza aplicada en una superficie específica. La presión se denota en diferentes unidades dependiendo del sistema, la presión se puede dar en: psi, Pascal, Atmósferas, centímetros de mercurio, etc.
- Para los fines prácticos de este capítulo se trabajará con el sensor de presión MPX4115, este sensor está caracterizado en Kilo Pascal, y permite medir presiones entre 15 y 115 KPa, o entre 2,18 y 16,7 psi.

Sensores de presión

- La apariencia física, la distribución de pines y la vista en ISIS, de este dispositivo es la siguiente:



<i>Pin</i>	<i>Función</i>
1	<i>Vout</i>
2	<i>GND, Referencia</i>
3	<i>Vs, Poder</i>
4, 5, 6	<i>No implementado</i>

Sensores de presión

- Independiente del sensor que se implemente, cada uno debe contar con una relación de salida o función de transferencia, en el caso particular del sensor MPX4115 la función de transferencia especificada por el fabricante es:

$$V_{out} = V_s(0,009P - 0,095)$$

- Donde V_{out} es el voltaje de salida del sensor, P es la presión en Kilo Pascal, y V_s el voltaje de alimentación del sensor. Despejando de la ecuación anterior, la presión P , se obtiene

$$P = \frac{111,111V_{out}}{V_s} + 10,555$$

- En función del convertidor AD, del PIC configurado con 10 bits de resolución se puede concluir que:

$$\frac{5V}{1023} = \frac{nV}{Radc}$$

Sensores de presión

- Para el caso particular de este ejemplo se usará una fuente de poder de 5 voltios la cual alimenta el sensor y el micro, por lo tanto V_s es igual a 5 voltios, y la relación final queda de la siguiente forma:

$$P = 0,10861 R_{adc} + 10,555$$



**Sergio
Velásquez**

Sensores de presión

- Para demostrar el funcionamiento práctico de este tipo de sensores se puede observar y analizar el siguiente código fuente:

```
//Definición de pines del LCD
```

```
sbit LCD_RS at RB4_bit;
```

```
sbit LCD_EN at RB5_bit;
```

```
sbit LCD_D7 at RB3_bit;
```

```
sbit LCD_D6 at RB2_bit;
```

```
sbit LCD_D5 at RB1_bit;
```

```
sbit LCD_D4 at RB0_bit;
```

```
//Definición de los TRIS del LCD
```

```
sbit LCD_RS_Direction at TRISB4_bit;
```

```
sbit LCD_EN_Direction at TRISB5_bit;
```

```
sbit LCD_D7_Direction at TRISB3_bit;
```

```
sbit LCD_D6_Direction at TRISB2_bit;
```

```
sbit LCD_D5_Direction at TRISB1_bit;
```

```
sbit LCD_D4_Direction at TRISB0_bit;
```

```
void main( void )
```

```
{
```

```
    //Declaración de variables.
```

```
    unsigned int Radc, PreI;
```

```
    float Pre;
```

```
    char Text[16];
```

```
    //Inicio del LCD.
```

```
    Lcd_Init();
```

```
    //Borrado del cursor.
```

```
    Lcd_Cmd( _LCD_CURSOR_OFF );
```

```
    //Impresión de texto.
```

```
    Lcd_Out( 1, 1, "Presion en KPa:");
```

```
    while(1) //Bucle infinito.
```

```
{
```

```
    //Lectura del canal 0 del ADC.
```

```
    Radc = ADC_Read(0);
```

```
    //Uso de la ecuación (13.9).
```

```
    Pre = 0.10861*Radc+10,5555;
```

```
    //Se fuerza el resultado a la parte entera.
```

```
    PreI = Pre;
```

```
    //Se convierte el número entero a una cadena de caracteres.
```

```
    IntToStr( PreI, Text );
```

```
    //Se imprime el resultado.
```

```
    Lcd_Out( 2, 1, Text);
```

```
    //Retardo de 100m segundos.
```

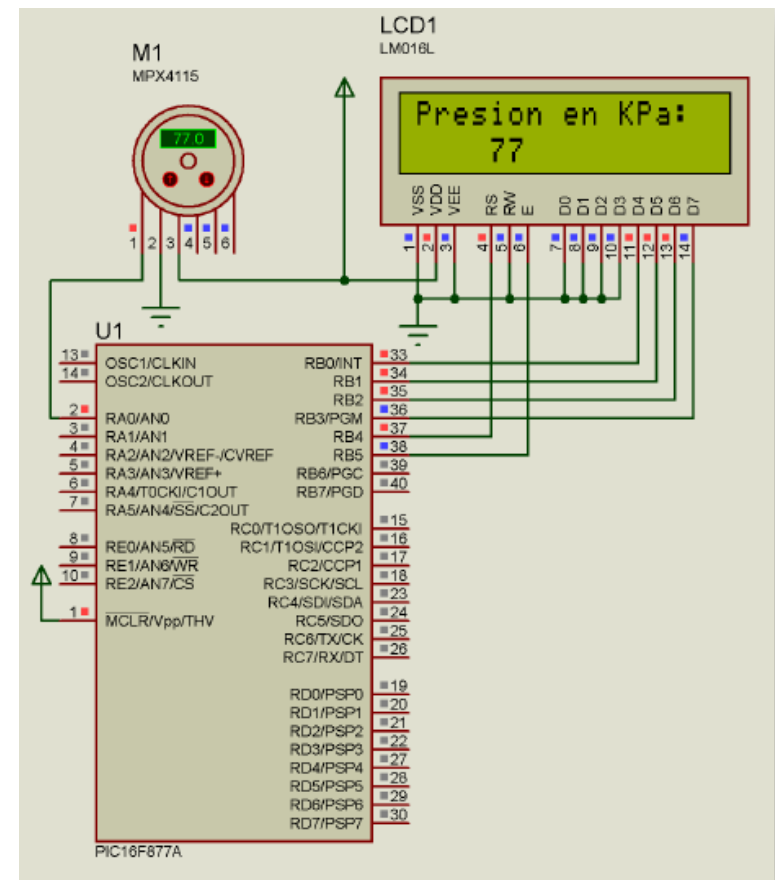
```
    delay_ms(100);
```

```
}
```

```
}
```

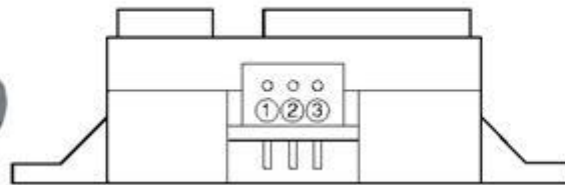
Sensores de presión

- Terminada la edición y compilación del programa se puede implementar en el simulador ISIS los dispositivos: 16F877A, MPX4115, y LM016L, para realizar la simulación del sistema, el circuito debe tener la siguiente apariencia:



Sensores de distancia

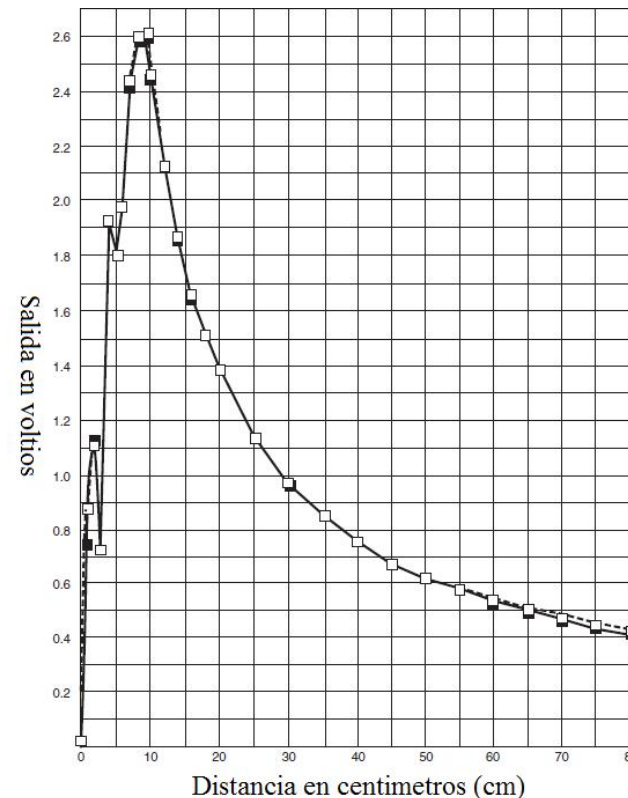
- Los sensores de distancia permiten medir una longitud desde el punto de ubicación del sensor hasta un obstáculo puesto en un punto dentro del rango de trabajo del sensor. La implementación de este tipo de dispositivos es de utilidad en sistemas de control, y en proyectos de robótica. Para el caso puntual de este capítulo se trabajará con el sensor GP2D12, este dispositivo usa como estrategia de medida un rayo de luz infrarroja y su reflejo para determinar la longitud. La apariencia física la distribución de pines y la vista en el simulador ISIS del sensor son los siguientes:



<i>Pin</i>	<i>Función</i>
1	<i>Vout</i>
2	<i>GND, Referencia</i>
3	<i>Vcc, Poder</i>

Sensores de distancia

- Este tipo de sensores cuentan con un comportamiento no lineal, esto quiere decir que la salida no obedece a una función de transferencia lineal. Para comprender de forma clara este concepto se puede observar y analizar la siguiente gráfica que muestra el comportamiento de la salida en función de la distancia:



Sensores de distancia

- Para iniciar este proceso se escogen 2 puntos, que estén equitativamente distribuidos. Por ejemplo el punto a 10cm, y 80cm. Para cada uno de los puntos se realiza una ecuación de orden igual al número de puntos menos uno, en este caso serán ecuaciones de primer orden. Se asume como variable independiente *x* los datos de entrada que en este caso son el voltaje de salida del sensor, como variable dependiente *y*, la distancia en centímetros. Para comprender este concepto observe la siguiente ecuación que representa el comportamiento exponencial del sensor:

$$y = \frac{A}{x} + B$$

Distancia(y)	Vo, Salida sensor(x)
10cm	2,35V
80cm	0,41V

- Sistema de ecuaciones:

$$10 = \frac{A}{2,35} + B$$

$$80 = \frac{A}{0,41} + B$$

- Sistema de ecuaciones con dos incógnitas:

$$10 = A0,425531 + B$$

$$80 = A2,43902 + B$$

Sensores de distancia

- Solucionando el sistema de ecuaciones se puede determinar que las constantes son:

$$A=34,76546392$$

$$B=-4,793814433$$

- De esta forma se puede formalizar la función de transferencia del sensor con la ecuación:

$$y = \frac{34,76546392}{x} + 4,793814433$$

- Como la variable x representa el voltaje de salida del sensor y teniendo presente que el valor del sensor no supera 2,5 voltios, la referencia del convertidor AD, se puede ajustar a 2,5 voltios. Para establecer la conversión se puede usar la ecuación

$$y = \frac{14226,02784}{Radc} - 4,793814433$$
$$Vadc = \frac{2,5Radc}{1023}$$

- El paso siguiente es implementar un código fuente que use la función de transferencia y que posteriormente visualice el valor capturado, para este fin se puede observar y analizar el siguiente código fuente:

Sensores de distancia

//Definición de pines del LCD

```
sbit LCD_RS at RB4_bit;  
sbit LCD_EN at RB5_bit;  
sbit LCD_D7 at RB3_bit;  
sbit LCD_D6 at RB2_bit;  
sbit LCD_D5 at RB1_bit;  
sbit LCD_D4 at RB0_bit;
```

//Definición de los TRIS del LCD

```
sbit LCD_RS_Direction at TRISB4_bit;  
sbit LCD_EN_Direction at TRISB5_bit;  
sbit LCD_D7_Direction at TRISB3_bit;  
sbit LCD_D6_Direction at TRISB2_bit;  
sbit LCD_D5_Direction at TRISB1_bit;  
sbit LCD_D4_Direction at TRISB0_bit;
```

void main(void)

```
{  
    //Declaración de variables.  
    unsigned int Radc, DisI;  
    float Dis;  
    char Text[16];  
    //Configura el módulo ADC con el pin AN3  
    //como voltaje de referencia positiva.  
    ADCON1 = 0b11000001;  
    //Inicio del LCD.  
    Lcd_Init();  
    //Borrado del cursor.
```

```
Lcd_Cmd( _LCD_CURSOR_OFF);
```

//Impresión de texto.

```
Lcd_Out( 1, 1, "Distancia:");
```

while(1) //Bucle infinito.

```
{
```

//Lectura del canal análogo.

```
Radc=ADC_Read(0);
```

//Implementación de la función de transferencia (13.15).

```
Dis = (14226.02784/Radc)-4.793814433;
```

//Se para el resultado a un valor entero.

```
DisI=Dis;
```

//Se convierte el valor entero en cadena de texto.

```
IntToStr( DisI, Text );
```

//Se imprime la lectura del sensor.

```
Lcd_Out( 2, 1, Text );
```

//Retardo de 100m segundos.

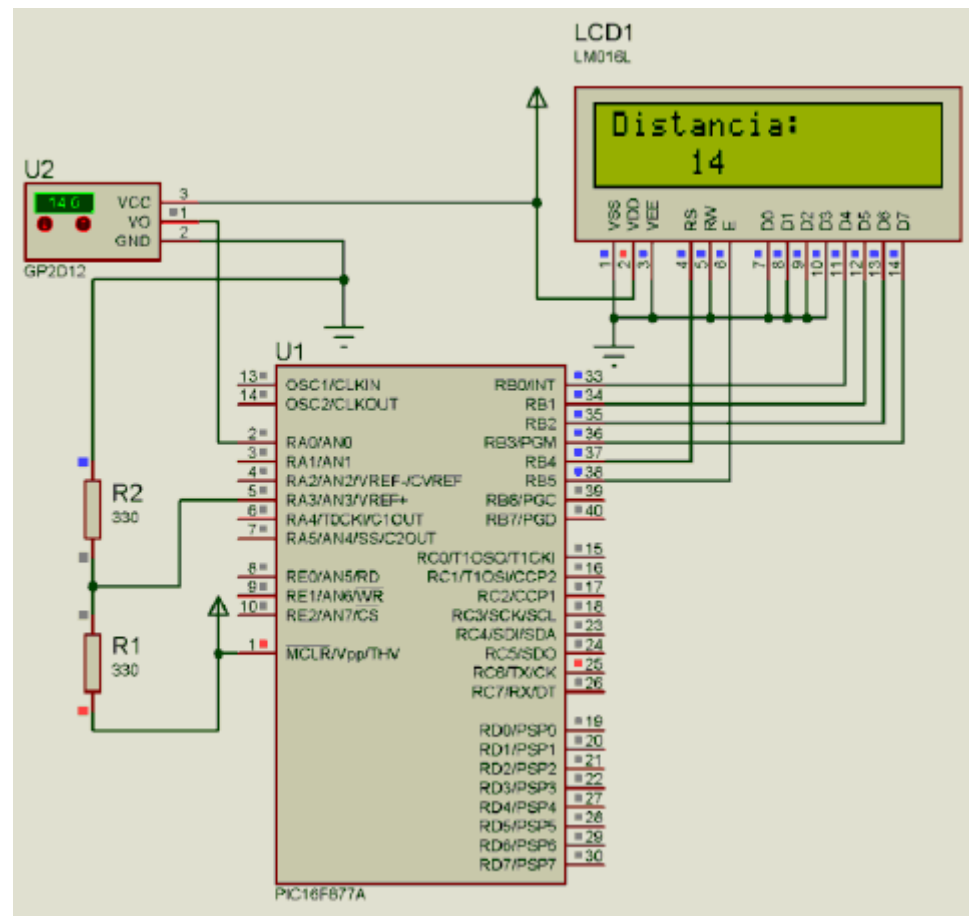
```
delay_ms(100);
```

```
}
```

```
}
```

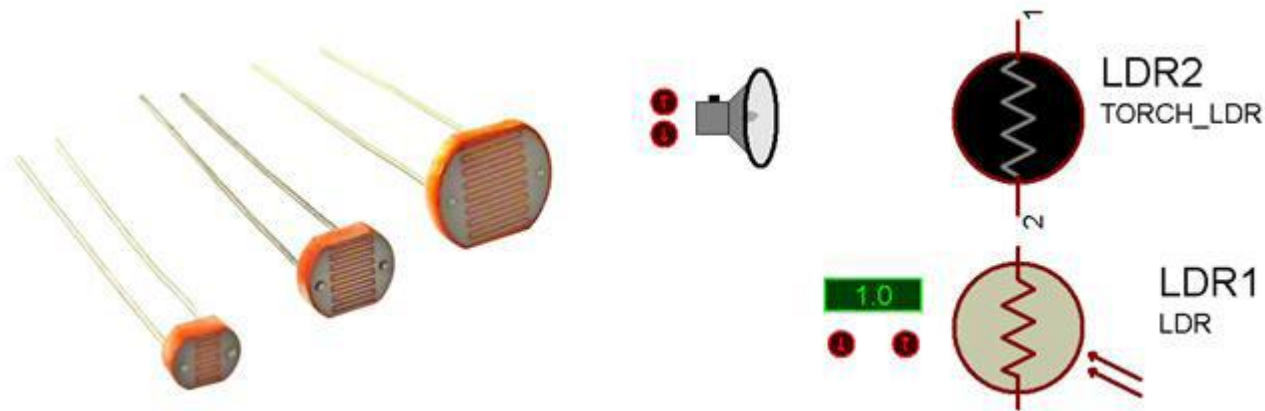

**Ser
Velás**

- 



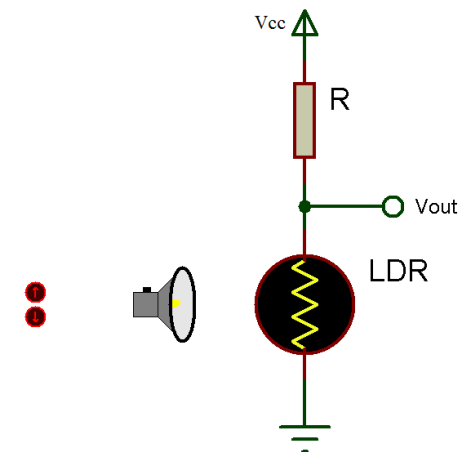
Sensores LDR

- Los sensores LDR, son dispositivos que cambian su resistencia en función de la intensidad de luz, cuanto mayor sea la intensidad de la luz, menor es la resistencia que ofrece la LDR. Estos dispositivos son útiles para determinar, la presencia o ausencia de luz en el ambiente, es posible concebir con estos dispositivos controles de luminosidad. La apariencia física y las vistas posibles en el simulador ISIS, son las siguientes:



Sensores LDR

- Las características eléctricas de estos dispositivos no son fáciles de conseguir dado su simple funcionamiento y gran variedad de fabricantes, las LDR, o fotoresistencias, se adquieren comercialmente en tamaños diversos, esto implica que el rango de resistencia cambia en función del tamaño, una LDR, de tamaño grande tiene rangos de cambio menor que una LDR pequeña. Incluso los cambios de resistencia no son exactamente iguales en dos LDR, del mismo tamaño. Estas razones hacen que la forma de mayor simplicidad para su uso sea la lectura de su resistencia, o el voltaje que se desempeña en sus terminales. Para usar una LDR, la forma más simple es realizar un divisor de voltaje con una resistencia fija, para entender este concepto se puede observar el siguiente arreglo o circuito:



Sensores LDR

- Asumiendo la teoría básica de circuitos eléctricos para un divisor de voltaje se puede implementar la siguiente ecuación:

$$LDR = \frac{(V_{out})(R)}{V_{cc} - V_{out}}$$

- Dado que el valor de salida V_{out} se puede ingresar a una entrada AD, del microcontrolador se puede asumir como V_{adc} , con la siguiente relación y un voltaje de referencia positivo de 5 voltios:

$$V_{adc} = \frac{(R_{adc})(5)}{1023} = (R_{adc})0,004887585$$

- Remplazando las ecuaciones, y asumiendo que V_{cc} es de 5 voltios, obtenemos la siguiente relación:

$$LDR = \frac{(R_{adc})(0,04887585)(R)}{5 - (R_{adc})(0,004887585)}$$

• Para el caso particular de la simulación de este ejercicio la resistencia del divisor de voltaje será de $10k\Omega$ de esta manera la ecuación para calcular la resistencia de la LDR, es:

$$LDR = \frac{(R_{adc})(48,87585)}{5 - (R_{adc})(0,004887585)}$$

Sensores LDR

- Para la compilación y edición del programa se puede observar y analizar el siguiente código fuente:

```
//Definición de pines del LCD
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D4 at RB0_bit;
//Definición de los TRIS del LCD
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D7_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB0_bit;
```

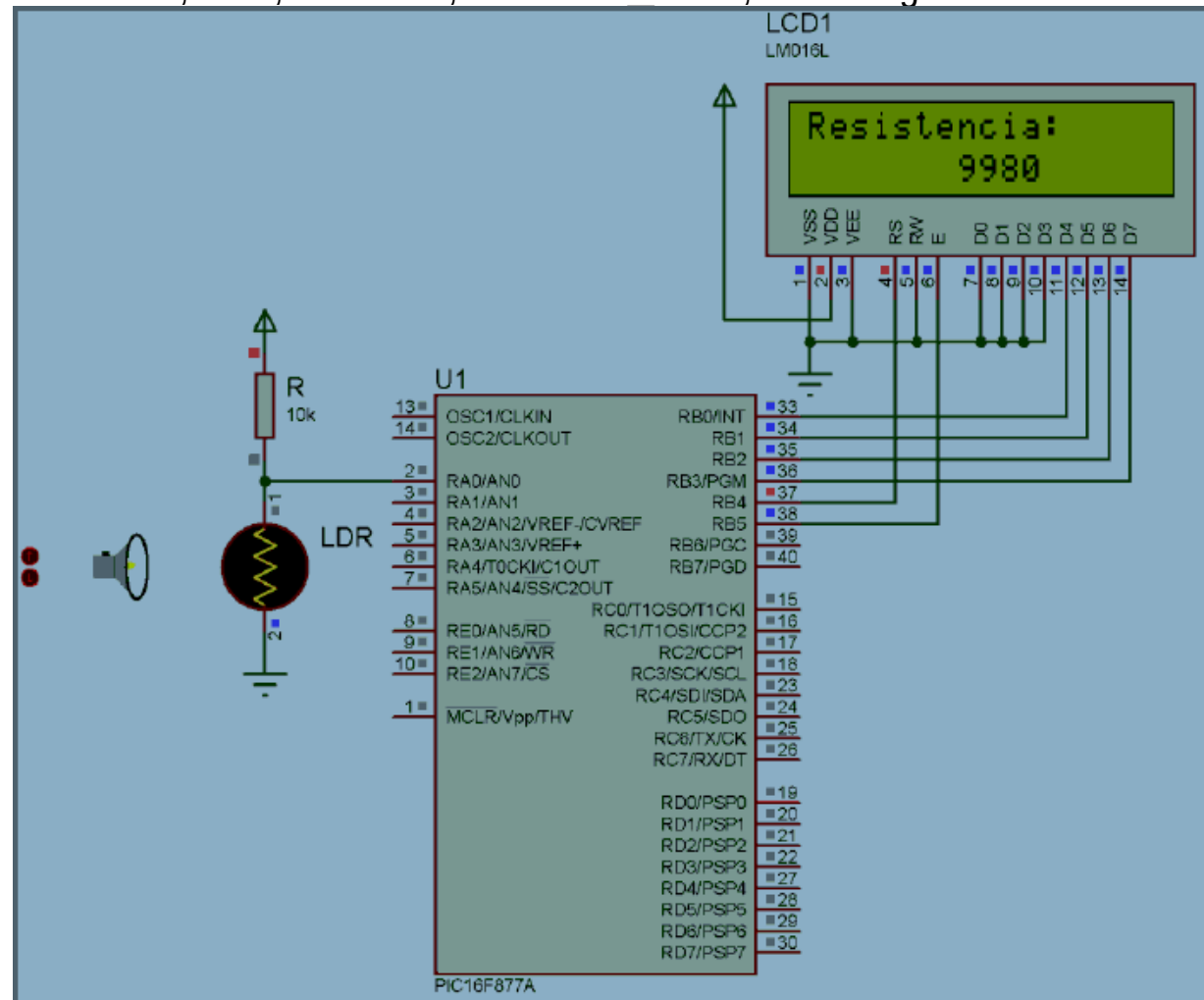
```
void main( void )
{
    //Declaración de variables.
    unsigned long Radc, DisI;
    char Text[16];
    //Inicio del LCD.
```

```
Lcd_Init();
//Borrado del cursor.
Lcd_Cmd(_LCD_CURSOR_OFF);
//Impresión de texto.
Lcd_Out( 1, 1, "Resistencia:");
while(1) //Bucle infinito.
{
    //Lectura del canal análogo.
    Radc=ADC_Read(0);
    DisI = (Radc*48.87585533)/(5.0-Radc*0.004887585);
    //Se convierte el valor entero largo, en cadena de texto.
    LongToStr( DisI, Text );
    //Se imprime la lectura del sensor.
    Lcd_Out( 2, 1, Text );
    //Retardo de 100m segundos.
    delay_ms(100);
}
```

```
}
```

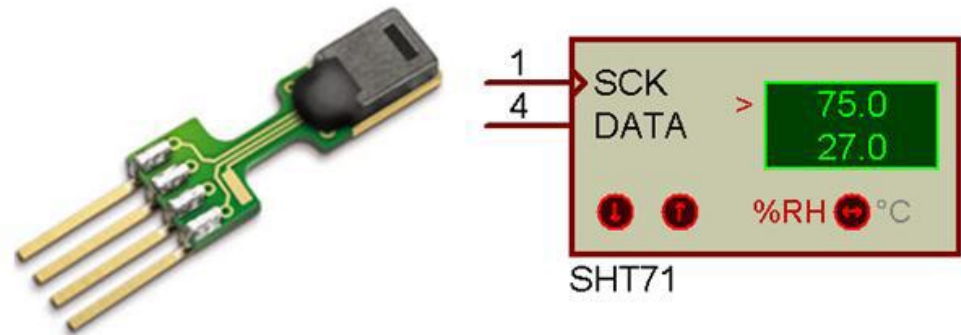
Sensores LDR

- Para realizar la simulación en ISIS, se implementan los siguiente dispositivos: 16F877A, RES, LM016L, TORCH LDR, en el siguiente circuito electrónico:



Sensores de humedad y temperatura

- La humedad y la temperatura son dos variables físicas de amplio uso en la industria y para su medida se pueden usar dispositivos que integran las dos lecturas simultáneamente, en un solo encapsulado. La familia de sensores SHT7x permiten hacer la lectura de ambos parámetros, con una calibración de fábrica que garantiza la fidelidad de las lecturas. La apariencia física y la vista de estos dispositivos en ISIS, así como su distribución de pines, es la siguiente:



Pin	Función
1	Reloj
2	Vdd, Poder
3	Gnd, Referencia
4	Datos

Sensores de humedad y temperatura

- Estos sensores entregan la información por medio de un puerto serial similar al protocolo I²C, sin embargo no es exactamente igual, y por esta razón no es posible usarlo. Esto implica realizar por medio del código de programa, el protocolo de comunicación especificado en la hoja técnica del dispositivo. El protocolo incluye una condición de inicio, bloque de datos tanto de lectura como de escritura, y un bit de confirmación o ACK. La comunicación se basa en dos pines uno de reloj que siempre debe ser de salida y un pin de datos bidireccional, el cual es de drenador abierto, lo que hace necesario implementar una resistencia pull-up en esta línea a Vcc de 10kΩ.
- Para realizar la comunicación con el sensor, el desarrollador debe implementar su propia librería para el enlace de datos, a continuación se mostrará paso a paso un ejemplo que permite cumplir con este cometido:

Sensores de humedad y temperatura

//Declaración de pines de comunicación.

sbit SHT_Reloj at RC0_bit;

sbit SHT_Datos at RC1_bit;

//Declaración de TRIS de los pines de comunicación.

sbit SHT_Reloj_Dire at TRISC0_bit;

sbit SHT_Datos_Dire at TRISC1_bit;

//Declaración de constantes.

const unsigned short RH_8bits_TEM_12bits = 1;

const unsigned short RH_12bits_TEM_14bits = 0;

const unsigned short Peso_bits[8] = {1,2,4,8,16,32,64,128};

//Bandera de resolución.

unsigned short RH_TEM;

//Función para establecer un estado

//lógico en el pin de datos

void SHT_Bus(**unsigned short** e)

{

//Si el estado es 1 se impone en alta

//impedancia el pin de datos.

if(e)

 SHT_Datos_Dire = 1;

else

{

//Si el estado es un 0 lógico

//se establece la salida en 0.

 SHT_Datos_Dire = 0;

 SHT_Datos = 0;

}

}



**Sergio
Velásquez**

Sensores de humedad y temperatura

//Función para establecer el estado del reloj.

void SHT_Reloj_Salida(unsigned short e)

```
{  
    //Se establece el estado lógico en el reloj.  
    if( e )  
        SHT_Reloj = 1;  
    else  
        SHT_Reloj = 0;  
    //Periodo del reloj, para frecuencia de 1MHz.  
    delay_us(1);  
}
```

//Función para crear la condición de inicio.

void SHT_Inicio(void)

```
{  
    //Secuencia de inicio.  
    SHT_Reloj_Salida(1);  
    SHT_Bus(0);  
    SHT_Reloj_Salida(0);  
    SHT_Reloj_Salida(1);  
    SHT_Bus(1);  
    SHT_Reloj_Salida(0);  
    delay_ms(1);  
}
```

//Función para escribir un dato en el bus.

void SHT_Escribe_Bus(unsigned short d)

```
{  
    unsigned short n;  
    //Bucle para el envío serial.  
    for( n=7; n!=255; n-- )  
    {  
        //Se asigna el bit según su peso.  
        if( (d>>n)&1 )  
            SHT_Bus( 1 );  
        else  
            SHT_Bus( 0 );  
        //Generación de pulsos de reloj.  
        SHT_Reloj_Salida( 1 );  
        SHT_Reloj_Salida( 0 );  
    }  
    //Generación de bit ACK.  
    SHT_Bus( 1 );  
    SHT_Reloj_Salida( 1 );  
    SHT_Reloj_Salida( 0 );  
}
```

Sensores de humedad y temperatura

//Función para leer un dato en el bus.

unsigned short SHT_Leer_Bus(void)

{

unsigned short n, Dato_SHT=0;

//Se configura el bus de entrada.

SHT_Bus(1);

//Bucle para recibir los bits de forma serial.

for(n=7; n!=255; n--)

{

//Pulso de reloj.

SHT_Relej_Salida(1);

//Se reciben los bits según su peso.

if(SHT_Datos)

Dato_SHT += Peso_bits[n];

SHT_Relej_Salida(0);

}

//Se transmite la condición de ACK.

SHT_Bus(0);

SHT_Relej_Salida(1);

SHT_Relej_Salida(0);

//Se deja el bus de entrada.

SHT_Bus(1);

//Se retorna el valor de la lectura.

return Dato_SHT;

}

void SHT_Status_Graba(unsigned short s)

{

//Se transmite el valor del status.

SHT_Inicio();

SHT_Escribe_Bus(0b00000110);

SHT_Escribe_Bus(s);

}

//Función para leer el valor del Status.

unsigned short SHT_Status_Leer(void)

{

unsigned short st;

SHT_Inicio();

SHT_Escribe_Bus(0b00000111);

st=SHT_Leer_Bus();

SHT_Leer_Bus();

return st;

}

Sensores de humedad y temperatura

//Función para leer la temperatura.

float SHT_Leer_Tem(void)

{

float tem;

unsigned int LEC=0;

unsigned short DH, DL;

//Se aplica el protocolo de lectura.

SHT_Inicio();

SHT_Escribe_Bus(0b00000011);

//Espera para hacer la lectura.

delay_ms(310);

DH=SHT_Leer_Bus();

DL=SHT_Leer_Bus();

SHT_Leer_Bus();

*LEC = DL + DH*256;*

//Se implementan las ecuaciones para

//la temperatura.

if(RH_TEM)

*tem = -40.1 + 0.04*LEC;*

else

*tem = -40.1 + 0.01*LEC;*

//Se retorna la temperatura.

return tem;

}

float SHT_Leer_Hum(void)

{

float hum;

//Se aplica el protocolo de lectura.

SHT_Inicio();

SHT_Escribe_Bus(0b00000101);

//Espera para hacer la lectura.

delay_ms(310);

DH=SHT_Leer_Bus();

DL=SHT_Leer_Bus();

SHT_Leer_Bus();

*LEC = DL + DH*256;*

//Se implementan las ecuaciones para

//la humedad.

if(RH_TEM)

*hum = -4.3468 + 0.5872*LEC - 0.00040845*LEC*LEC;*

else

*hum = -4.3468 + 0.0367*LEC - 0.0000015955*LEC*LEC;*

//Se retorna la humedad.

return hum;

}

Sensores de humedad y temperatura

//Función para inicializar el sensor.

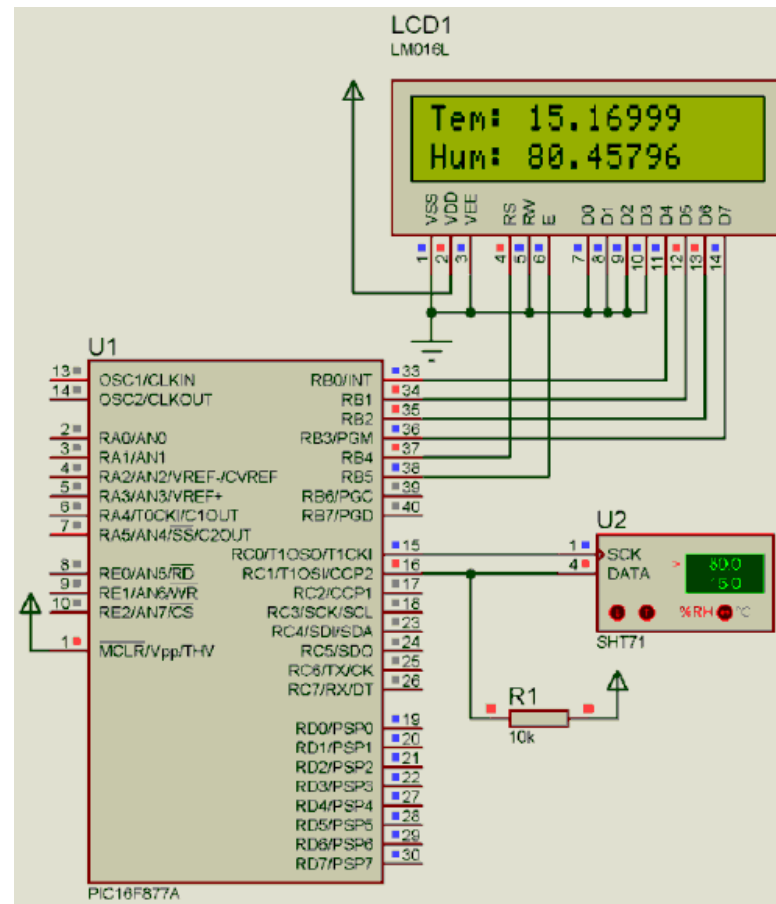
```
void Inicio_SHT7x( unsigned short St )
{
    unsigned short n;
    //Configuración de pines,
    //y estado inicial del bus de datos.
    SHT_Reloj_Dire = 0;
    //Se establece el pin de datos en 1.
    SHT_Bus(1);
    //Se inicia el reloj en 0.
    SHT_Reloj_Salida(0);
    delay_ms(100);
    //Se generan 9 ciclos de reloj, para reiniciar el sensor.
    for(n=0; n<9; n++)
    {
        SHT_Reloj_Salida(1);
        SHT_Reloj_Salida(0);
    }
    //Se configura el registro Status.
    SHT_Inicio();
    SHT_Status_Graba( St );
    delay_ms(1);
    RH_TEM = St;
}
```

Para el caso puntual de este ejemplo se complementa el código con la siguiente función main:

```
void main( void )
{
    //Declaración de variables.
    float Sen;
    char Text[16];
    //Inicio del LCD.
    Lcd_Init();
    //Se inicializa el sensor SHT7x.
    Inicio_SHT7x(RH_12bits_TEM_14bits);
    //Borrado del cursor.
    Lcd_Cmd( _LCD_CURSOR_OFF );
    //Impresión de texto.
    Lcd_Out( 1, 1, "Tem:" );
    Lcd_Out( 2, 1, "Hum:" );
    while(1) //Bucle infinito.
    {
        //Se lee la temperatura y se imprime en el LCD.
        Sen = SHT_Leer_Tem();
        FloatToStr( Sen, Text );
        Lcd_Out( 1, 6, Text );
        //Se lee la humedad y se imprime en el LCD.
        Sen = SHT_Leer_hum();
        FloatToStr( Sen, Text );
        Lcd_Out( 2, 6, Text );
    }
}
```

Sensores de humedad y temperatura

- Finalizada la edición y compilación del código fuente, se procede a realizar la simulación en ISIS, con los dispositivos: 16F877A, LM016L, RES, y SHT71, en el siguiente circuito electrónico:



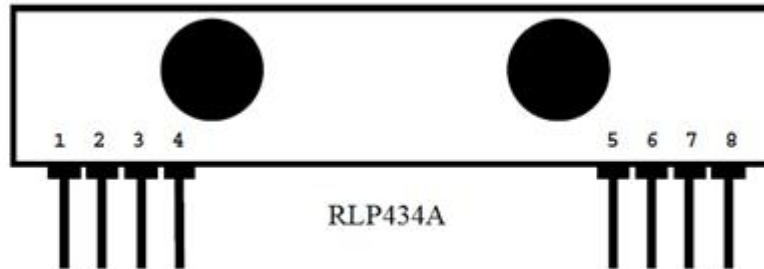
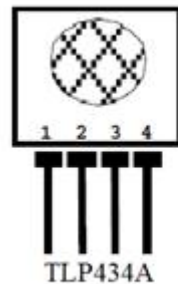
Módulos inalámbricos unidireccionales

- Las comunicaciones inalámbricas entre dos puntos hacen parte fundamental de muchas aplicaciones en el ámbito del control y el monitoreo de variables o estados. La lectura de un sensor, o el control de actuadores son posibles por medio de enlaces inalámbricos. La comunicación usada por estos dispositivos es unidireccional o simplex. Los módulos inalámbricos útiles para este fin funcionan con modulación ASK, o FSK, y tienen velocidades de transmisión hasta 9600 bits por segundo. Este tipo de dispositivos se pueden conseguir de fabricantes como Laipac Tech, o Linx Technologies, entre otros. La apariencia física de estos dispositivos es la siguiente:



Módulos inalámbricos unidireccionales

- Estos dispositivos solo son un medio de transmisión inalámbrico por lo cual no tienen ningún sistema de corrección ni detección de errores de transmisión, por esta razón es importante hacer dentro de la programación alguna rutina de detección de errores. Para fines de este ejemplo se implementarán los radios Laipac. De igual manera las frecuencias portadoras de estos dispositivos son variadas, y funcionan dentro del espectro de bandas libres para enlaces de comunicación. La distribución de pines de estos dispositivos es la siguiente



TLP434A	
PIN	Función
1	GND, Referencia
2	Entrada serial
3	Vcc, de 5 a 12 V
4	Antena

RLP434A	
PIN	Función
1	GND, Referencia
2	Salida Serial
3	Salida de prueba
4	Vcc
5	Vcc 5 V
6	GND, Referencia
7	GND, Referencia
8	Antena

Módulos inalámbricos unidireccionales

- Para el envío de bits este ejemplo usa la modulación por ancho de pulso o PWM. El programa representa un 1 lógico con un pulso en alto de 2m segundos de duración seguido por 1m segundo en bajo, el 0 lógico se transmite con un pulso en alto de 1m segundo y 1m segundo en bajo, y como estrategia de sincronismo se implementa un pulso en alto de 500u segundos, y 500u segundos en bajo. Para este ejemplo se envían 16 pulsos de sincronismo, posteriormente se envían 8 bits con el byte que se desea transmitir, y por último se transmiten 4 bits con un número de 0 a 8 que contiene el número de unos que contiene el byte que se está transmitiendo como estrategia de verificación. Este sistema no está en capacidad de corregir errores de transmisión, pero si permite verificar el dato recibido, en otras palabras el sistema entrega un dato que es recibido correctamente y si el dato se detecta con errores se descarta. Cuando un dato es recibido correctamente se entrega un byte que puede valer de 0 a 255, y si el dato está dañado se entrega un valor igual a -1.

Módulos inalámbricos unidireccionales

- Para la demostración de esta comunicación se implementará una librería que transmite bytes con las reglas antes mencionadas, de igual manera los microcontroladores implementados en esta librería son los PIC 16F628A con frecuencia de reloj de 4M Hz, sin embargo puede ser usada en cualquier otro microcontrolador, la librería básica es la siguiente:

//Declaración de pines de comunicación.

sbit RXPIN at RB7_bit;

sbit RXPIN_TRIS at TRISB7_bit;

sbit TXPIN at RB6_bit;

sbit TXPIN_TRIS at TRISB6_bit;

//Declaración de constantes de peso binario.

const unsigned int Peso[12] = {1,2,4,8,16,32,64,128,256,512,1024,2048};

void Inicio_Radio_Tx(**void**)

{

TXPIN_TRIS=0;

TXPIN=0;

}

//Función para iniciar los pines de comunicación.

void Inicio_Radio_Rx(**void**)

{

RXPIN_TRIS=1;

}

Módulos inalámbricos unidireccionales

//Función para determinar que bit se está recibiendo.

char Entra_Bit(void)

```
{  
    unsigned short T=0;  
    while( !RXPIN ); //Se espera el estado 1 lógico.  
    //Se mide el tiempo en alto de la modulación.  
    while( RXPIN ){ delay_us(50); T++; }  
    //Se clasifica el bit recibido en función de su tiempo.  
    if( T > 38 && T < 42 )return '1'; //Aprox. 2m Seg.  
    if( T > 18 && T < 22 )return '0'; //Aprox. 1m Seg.  
    if( T > 8 && T < 12 )return 'S'; //Aprox. 500u Seg.  
    return 'E'; // Tiempo errado.  
}
```

//Función para enviar un byte.

int Resibir_Dato(void)

```
{  
    unsigned short D, BIT, N_Bits;  
    int DATO;  
    DATO=0;  
    BIT=0;  
    //Se espera un bit de sincronismo.  
    while( Entra_Bit()!='S' );  
    //Se espera el último BIT de sincronismo.  
    while( (D=Entra_Bit())=='S' );  
    //Bucle para archivar el byte de entrada.  
    while( 1 )  
    {  
        //Se evalúa el BIT que llego.  
        switch( D )  
        {  
            case '1': DATO += Peso[BIT++];  
                break;  
            case '0': BIT++;  
                break;  
            case 'E':  
            case 'S': return -1;  
        }  
    }  
}
```



**Sergio
Velásquez**

Módulos inalámbricos unidireccionales

```
}  
//Si el número bits recibidos es 12, se  
//evalúan los errores.  
if( BIT==12 )  
{  
    //Se cuentan los bits del dato de información.  
    N_Bits=0;  
    for( BIT=0; BIT<8; BIT++ )  
        if( (DATO>>BIT)&1 )N_Bits++;  
    //Se verifica que el número de unos sea igual.  
    //y se retorna el byte de entrada.  
    if( N_Bits == (DATO>>8)&0x0F )return DATO&0x00FF;  
    //Ocurre un error y se espera una nueva trama.  
    return -1;  
}  
//Se lee el siguiente bit.  
D=Entra_Bit();  
}
```

```
//Función para transmitir un byte.  
void Enviar_Dato( unsigned short d )  
{  
    unsigned short BITS, N_Bits=0;  
    //Se cuentan el número de bits a enviar.  
    for( BITS=0; BITS<8; BITS++ )  
        if( (d>>BITS)&1 )N_Bits++;  
  
    //Se envían 16 bits de sincronismo.  
    for( BITS=0; BITS<16; BITS++ )  
    {  
        TXPIN = 1; delay_us(500);  
        TXPIN = 0; delay_us(500);  
    }  
  
    //Se envían 8 bits de datos, el byte.  
    for( BITS=0; BITS<8; BITS++ )  
    {  
        TXPIN=1;  
        if( (d>>BITS)&1 )  
            delay_ms(2);  
        else  
            delay_ms(1);  
        TXPIN=0;  
        delay_ms(1);  
    }  
}
```

The logo consists of the letters 'SV' in a bold, red, sans-serif font, centered within a thin black circular outline.

**Sergio
Velásquez**

Módulos inalámbricos unidireccionales

```
//Se envían 4 bits de verificación
//con el número de unos en el byte.

for( BITS=0; BITS<4; BITS++ )
{
    TXPIN=1;
    if( (N_Bits>>BITS)&1 )
        delay_ms(2);
    else
        delay_ms(1);
    TXPIN=0;
    delay_ms(1);
}
//Retardo final.
delay_ms(3);
}
```

Para concretar el ejemplo se realizan dos programas, uno para la transmisión, y otro para la recepción, este sistema transmite el estado de 4 pulsadores y se puede visualizar en el receptor por medio de 4 LEDs. De igual manera la simulación puede soportar los dos microcontroladores al mismo tiempo, pero no puede simular los radios. Para fines de simulación se implementa una conexión directa, sin embargo el desarrollador puede posteriormente hacer una prueba real con los radios.

Cada uno de los programas debe tener anexa la librería anteriormente citada, y anexar las siguientes funciones main, en cada uno de los códigos fuente:

Módulos inalámbricos unidireccionales

Función main para el transmisor:

```
void main( void )  
{  
    //Declaración de variables.  
    unsigned char DATO;  
    //Configuración de puertos.  
    TRISB = 0xFF;  
    //Activación de las resistencias PULL-UP del puerto B.  
    OPTION_REG = 0;  
    //Inicio del puerto de transmisión.  
    Inicio_Radio_Tx();  
  
    while(1) //Bucle infinito.  
    {  
        //Adquisición de estados de los pulsadores.  
        DATO = (~PORTB)&0x0F;  
        //Transmisión del byte.  
        Enviar_Dato( DATO );  
    }  
}
```

Módulos inalámbricos unidireccionales

Función main para el receptor:

```
void main( void )
{

    //Declaración de variables.
    int DATO;
    //Configuración de puertos.
    TRISB = 0;
    PORTB = 0;
    //Inicio del puerto de recepción.
    Inicio_Radio_Rx();

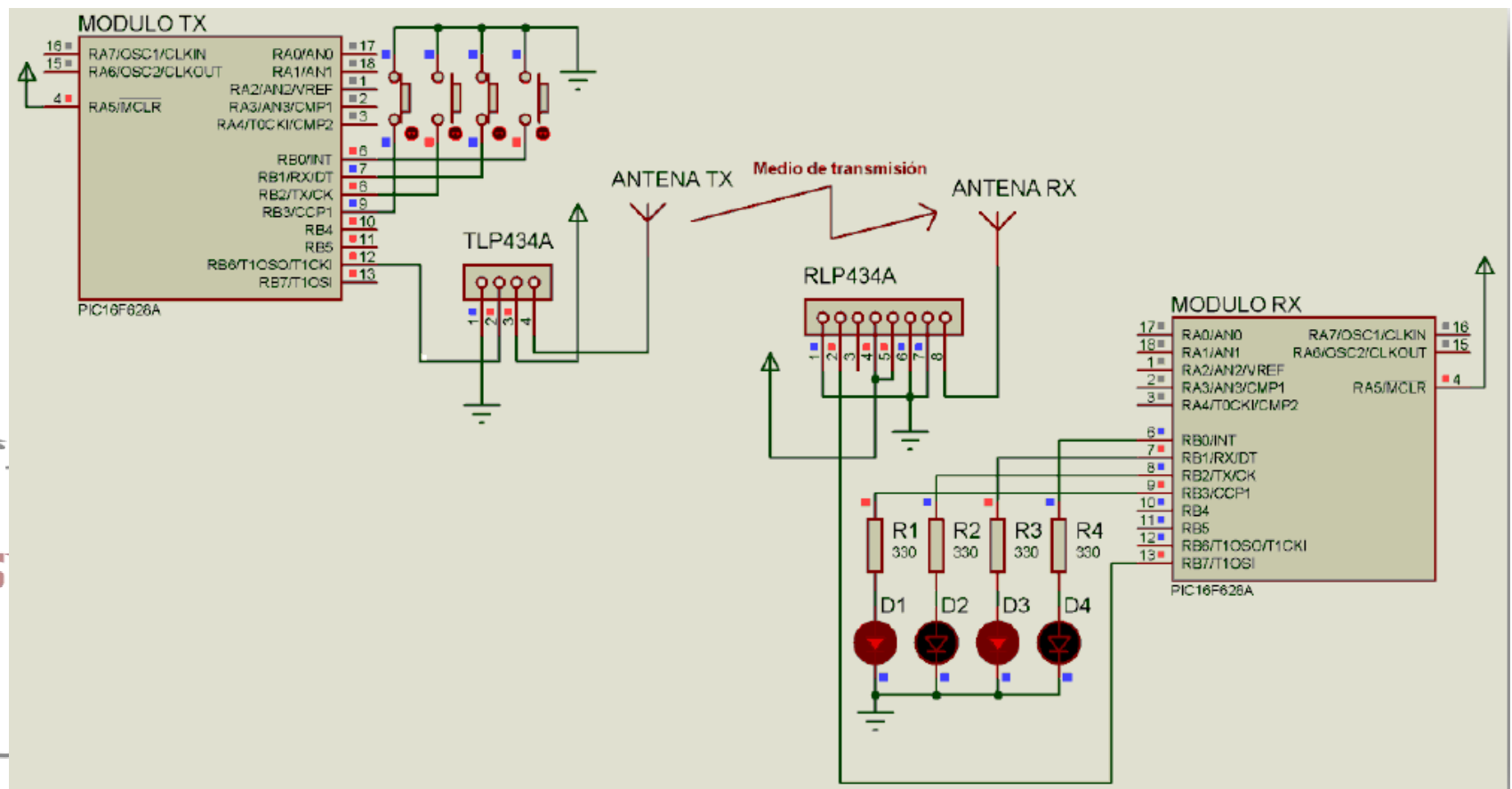
    while(1) //Bucle infinito.
    {
        //Lectura del byte de entrada.
        DATO = Resibir_Dato();
        //Verificación de errores, y visualización del byte.
        if( DATO!=-1 )
            PORTB =DATO;
    }
}
```

- 
- PIC16F628A
- Sergio Velásquez**



Módulos inalámbricos unidireccionales

- Para fines prácticos con los radios Laipac, se debe hacer un par de circuitos como los que se pueden ver en la siguiente gráfica:



Módulos inalámbricos bidireccionales

- En algunos casos las comunicaciones deben ser imperativamente bidireccionales, como el caso de los controles realimentados y los sistemas de comunicación full duplex. Para cumplir con el propósito de realizar la comunicación full duplex se pueden implementar módulos de radio prediseñados como los XBee, o XBee PRO, estos dispositivos son módulos MODEM, que trabajan sobre la banda de 2,4GHz, e implementan el estándar IEEE 801.15.4. Estos son dispositivos que trabajan con tecnología de 3 voltios de bajo consumo, y velocidades de transmisión de 1200 a 115200 bits por segundo. El alcance de los radios depende de la antena y de la referencia de los mismos. Los módulos XBee tienen un alcance de 100 metros con una línea de vista directa y 1000 metros en los módulos XBee PRO. Los módulos XBee, son de fácil instalación y programación por medio de un paquete de software, suministrado por el fabricante, este se denomina: X-CTU, este aplicativo se puede descargar gratuitamente en la página: www.digi.com, este programa establece con el radio la comunicación para su programación por medio de un puerto serial COM.

The logo consists of the letters 'SV' in a bold, red, sans-serif font, enclosed within a thin black circular border. The background of the slide features a light blue vertical band on the left with a pattern of overlapping circles and a grid of small dots.

SV

Sergio
Velásquez

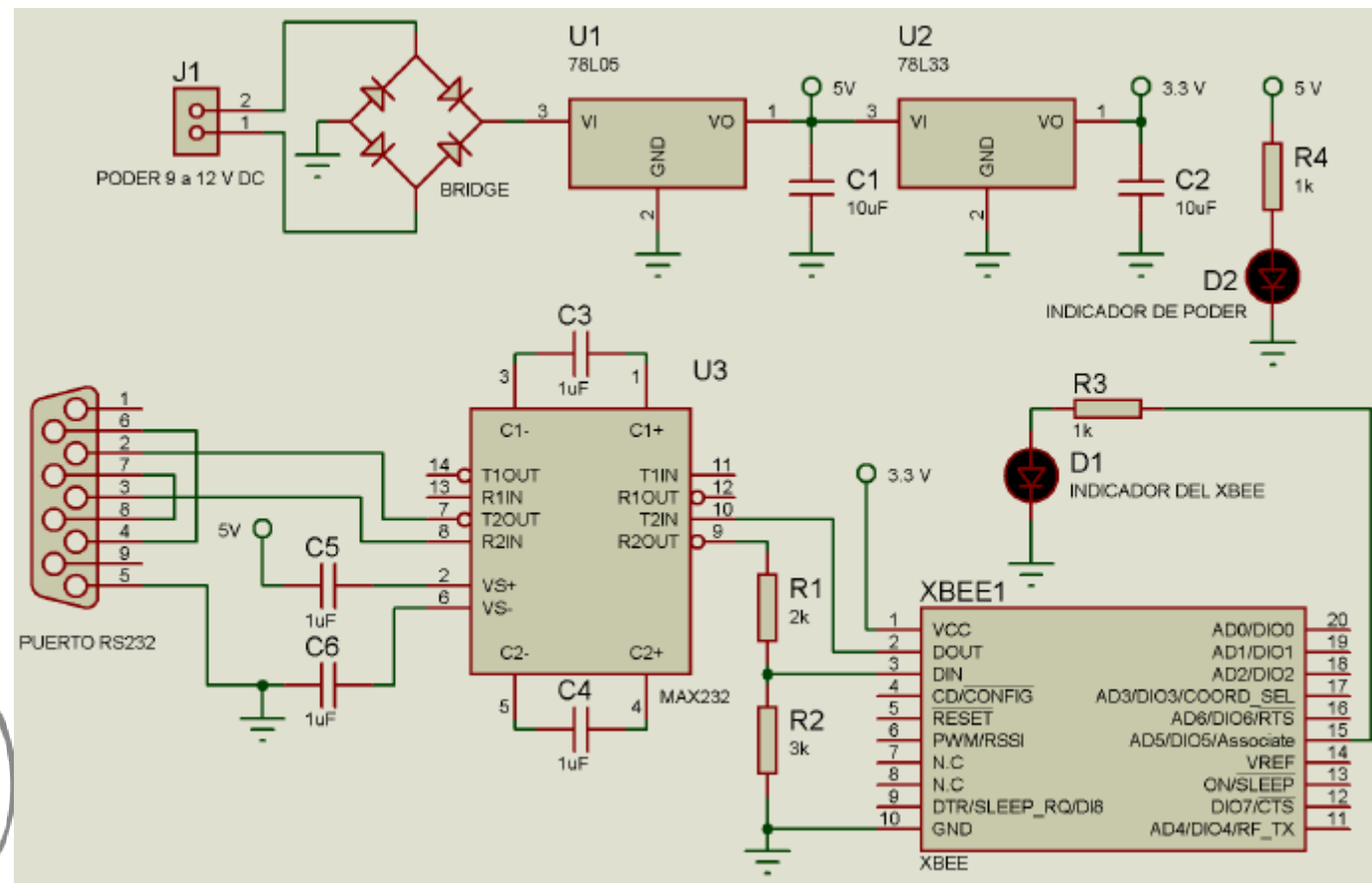
Módulos inalámbricos bidireccionales

- La simulación de estos radios no es posible con ISIS; pero dado el óptimo desempeño de los mismos es equivalente en ISIS a realizar una conexión cruzada entre Tx y Rx, de los microcontroladores. Los módulos XBee incorporan dentro de su hardware y software un procedimiento de detección y corrección de errores en la transmisión. De la misma forma es posible establecer una conexión inalámbrica entre un microcontrolador con un ordenador personal por medio de un puerto serial COM. Para trabajar y programar los radios XBee es necesario crear o implementar un acople de hardware para hacer la conexión con un puerto serial COM, para este proposito se debe tener presente el siguiente circuito electrónico:



**Sergio
Velásquez**

Módulos inalámbricos bidireccionales



Módulos inalámbricos bidireccionales

- Los módulos XBee por defecto vienen programados para establecer una comunicación a 9600 bits por segundo. Sin embargo se puede cambiar por medio del programa X-CTU. La reprogramación del radio XBee, permite cambiar los parámetros de velocidad, canal de comunicación entre 16 posibles, se puede asignar un identificador del radio y asignar dirección de trabajo en una red Zigbee.
- Para hacer una demostración del funcionamiento con los módulos XBee, se puede editar y analizar el siguiente código fuente:

Módulos inalámbricos bidireccionales

//Declaración de pines de control para LCD.

```
sbit LCD_RS at RB4_bit;  
sbit LCD_EN at RB5_bit;  
sbit LCD_D4 at RB0_bit;  
sbit LCD_D5 at RB1_bit;  
sbit LCD_D6 at RB2_bit;  
sbit LCD_D7 at RB3_bit;  
sbit LCD_RS_Direction at TRISB4_bit;  
sbit LCD_EN_Direction at TRISB5_bit;  
sbit LCD_D4_Direction at TRISB0_bit;  
sbit LCD_D5_Direction at TRISB1_bit;  
sbit LCD_D6_Direction at TRISB2_bit;  
sbit LCD_D7_Direction at TRISB3_bit;
```

void main(void)

```
{  
    //Declaración de variables.  
    char Texto[30];  
    char Dato;  
    unsigned short N=0;  
    //Inicio de los módulos USART, y LCD.  
    Lcd_Init();  
    UART1_Init(9600);
```

Módulos inalámbricos bidireccionales

```
//Se imprimen los textos de inicio.
Lcd_Out(1,1,"Datos de entrada:");
Lcd_Out(2,1,"");
while(1) //Bucle infinito.
{
    //Se evalúa si un dato llega por el puerto serial.
    if( UART1_Data_Ready() )
    {
        //Se lee el dato de entrada.
        Dato = UART1_Read();
        //Se evalúa que Dato llega por el puerto serial.
        switch( Dato )
        {
            //Se evalúa la llegada de un enter.
            case 13: N=0;
                //Se limpian las líneas del LCD.
                Lcd_Out(2,1,""); //Se imprimen 20 espacios vacíos.
                Lcd_Out(2,1,""); //Se imprimen 20 espacios vacíos.
                Lcd_Out(4,1,""); //Se imprimen 20 espacios vacíos.
                //Se posiciona el cursor en la segunda línea.
                Lcd_Out(2,1,"");
                //Se transmite un enter y un retroceso del carro.
                UART1_Write(13);
                UART1_Write(10);
                //Se transmite el texto: Texto Xbee.
                UART1_Write_Text("Texto Xbee: ");
                //Se retransmite la cadena recibida.
                UART1_Write_Text(Texto);
                //Se transmite un enter y un retroceso del carro.
                UART1_Write(13);
                UART1_Write(10);
                break;
```

Módulos inalámbricos bidireccionales

default: //Se guardan los datos entrantes en la cadena de caracteres Texto.

Texto[N++] = Dato;

Texto[N] = 0;

//Se refresca la información en el LCD.

Lcd_Out(2,1,Texto);

//Se retransmite eco del dato de entrada.

UART1_Write(Dato);

break;

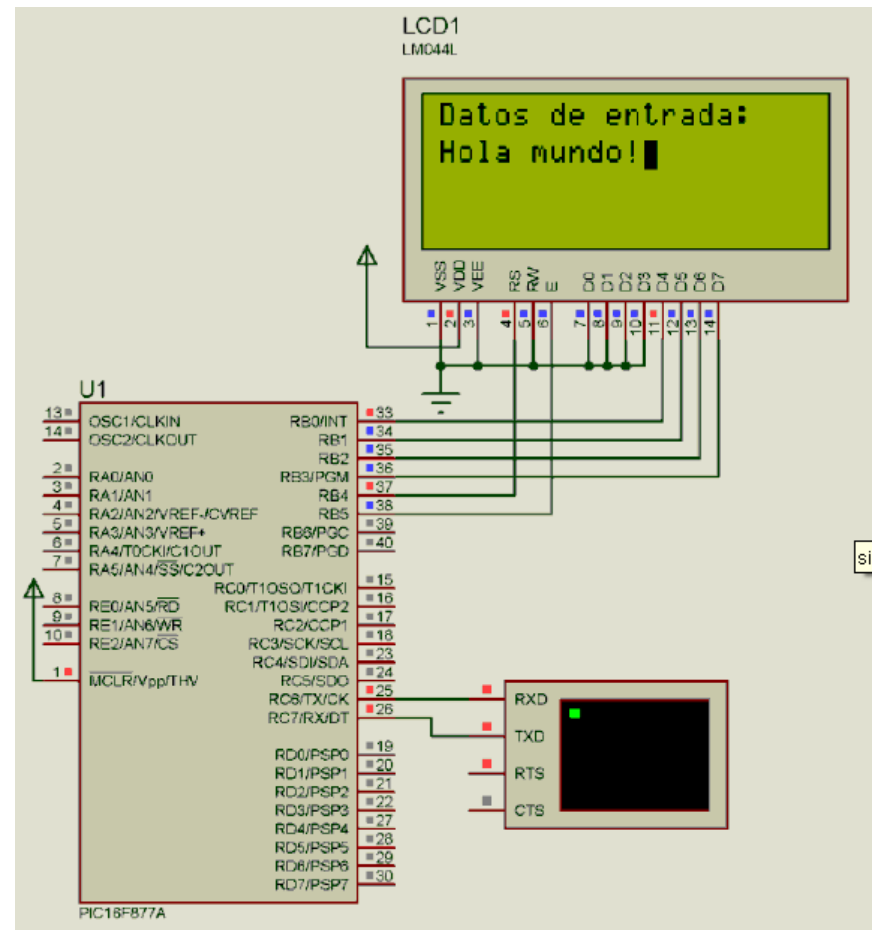
}
}
}
}



**Sergio
Velásquez**

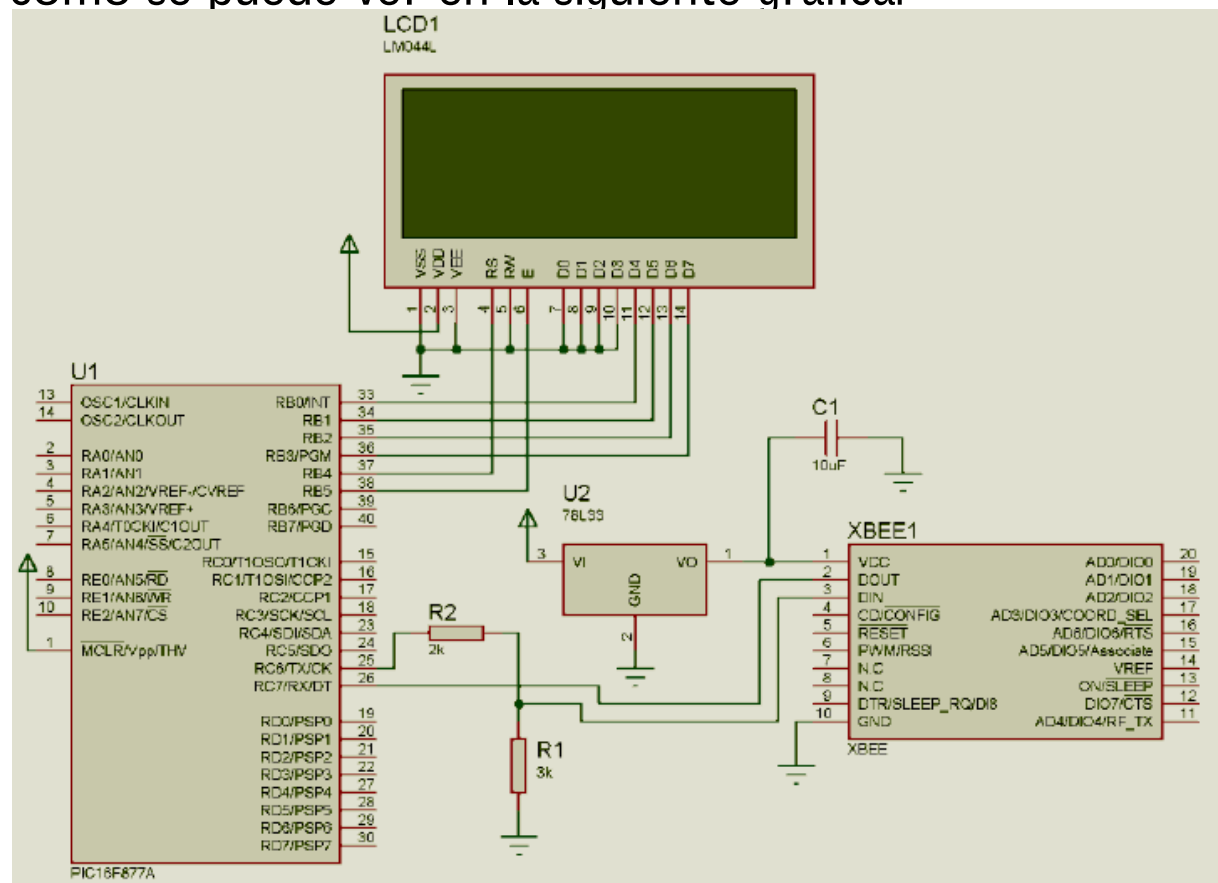
Módulos inalámbricos bidireccionales

- Para fines de simulación se puede implementar un circuito con los dispositivos: 16F877A, LM044L, y el puerto virtual VIRTUAL TERMINAL, como se puede apreciar en el siguiente esquema:



Módulos inalámbricos bidireccionales

- Para fines prácticos se puede usar el módulo de acople con un ordenador personal, antes citado con una sesión de Hyper terminal de Windows. Como parte complementaria se puede hacer un circuito real como se puede ver en la siguiente gráfica:



Módulos inalámbricos infrarrojos

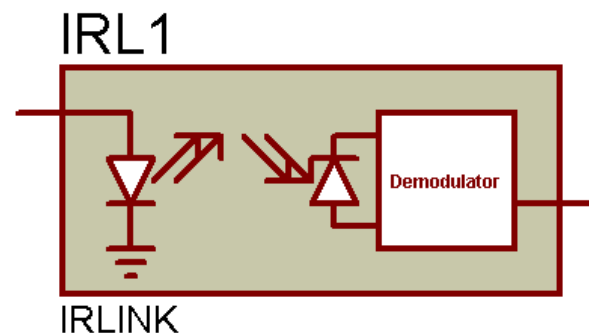
- La comunicación infrarroja es una estrategia práctica para sistemas que no requieren una gran velocidad de transmisión, y que admiten línea de vista directa. El alcance de un enlace infrarrojo depende de la potencia del emisor. Sin embargo este estándar es utilizado en la comunicación de control de los electrodomésticos convencionales como: televisores, equipos de sonido, reproductores de video, aires acondicionados entre otros, tiene un alcance promedio de 10 metros.
- La transmisión de datos se realiza por medio de diodos emisores de luz infrarroja, con una modulación digital por amplitud, ASK. Los estándares de comunicación para los controles infrarrojos implementan frecuencias portadoras de 36, 38, y 40 K Hz entre otras. La recepción de los datos se hace por medio de un demodulador ASK, que puede recibir datos hasta los 2,4K bits por segundo. Los demoduladores ASK, infrarrojos son dispositivos de carácter integrado que se pueden conseguir comercialmente. Estos poseen tres terminales que corresponden a dos de polarización, Vcc o poder, y referencia, y una salida demodulada en amplitud, con colector abierto, esto quiere decir que se requiere el uso de una resistencia PULL-UP, a Vcc, en este último terminal. Algunos fabricantes incorporan esta resistencia en la integración del dispositivo.

Módulos inalámbricos infrarrojos

- Para fines de diseño es importante tener presente que la salida del demodulador está invertida, es decir que cuando el dispositivo detecta la señal portadora la salida del demodulador se activa en bajo y se activa en alto cuando se detecta la ausencia de la misma. La apariencia física de estos dispositivos es la siguiente:



- La distribución de pines puede variar de un fabricante a otro, sin embargo siempre serán las mismas tres, Vcc, Referencia, y Salida, para fines prácticos se hace imperativo consultar la ficha técnica del dispositivo que se use.
- Para la simulación de este tipo de enlaces se puede usar en ISIS el módulo virtual: IRLINK, el cual tiene la siguiente apariencia en el simulador



Módulos inalámbricos infrarrojos

- Este dispositivo virtual simula el enlace inalámbrico por medio del LED infrarrojo, y el dispositivo demodulador, de igual manera permite editar la frecuencia de la señal portadora ASK. Para contextualizar el desempeño de estos enlaces se debe observar y analizar el siguiente código fuente, que usa el módulo PWM, como fuente portadora, y el módulo USART, como señal moduladora, la información transmitida son datos ASCII, con textos fijos que se pueden ver en un hyper terminal virtual:

```
void main( void )  
{  
    //Configuración del módulo USART a 2400 bps.  
    UART1_Init(2400);  
    //Configuración de PWM para crear la frecuencia portadora a 38K Hz.  
    PWM1_Init(38000);  
    PWM1_Start();  
}
```



**Sergio
Velásquez**

Módulos inalámbricos infrarrojos

//Se configura el ciclo útil del PWM al 50%.

PWM1_Set_Duty(127);

while(1) //Bucle infinito.

{

//Se transmite información en un texto fijo.

UART1_Write_Text("Transmision de datos infrarrojos.");

//Se envía los caracteres enter y retroceso del carro.

UART1_Write(13);

UART1_Write(10);

//Se transmite información en un texto fijo.

UART1_Write_Text("Está emision está modulada a 38K Hz.");

UART1_Write(13);

UART1_Write(10);

//Se transmite información en un texto fijo.

UART1_Write_Text("Con una velocidad de transmision de 2400 bps.");

//Se envía los caracteres enter y retroceso del carro.

UART1_Write(13);

UART1_Write(10);

//Se envía los caracteres enter y retroceso del carro.

UART1_Write(13);

UART1_Write(10);

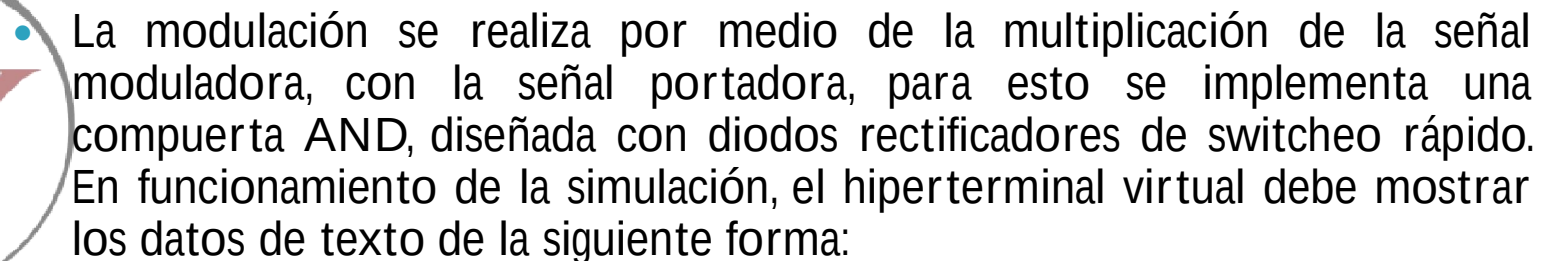
//Se realiza una pausa de 200m segundos.

delay_ms(200);

}

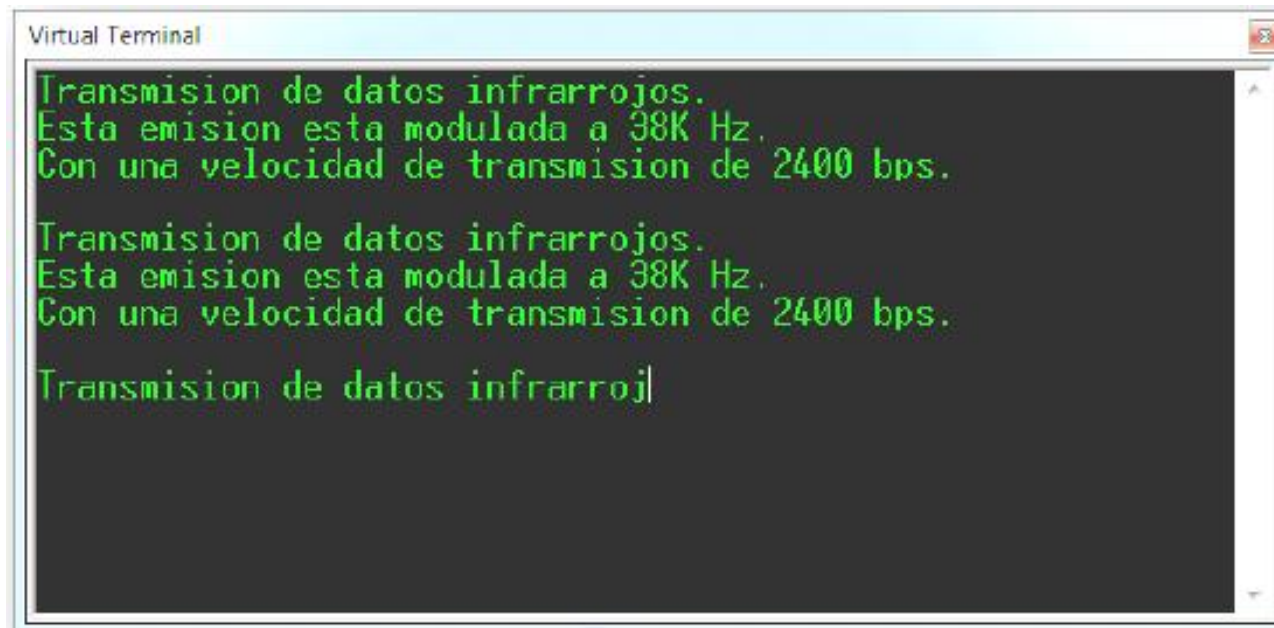
}

- 
- La
mo
co
En
los
- Sergio
Velásquez**



Módulos inalámbricos infrarrojos

- Para la simulación de este ejemplo se implementan en ISIS los dispositivos: 16F628A, RES, 1N4148, IRLINK, 74LS04, y el VIRTUAL TERMINAL. El circuito electrónico debe ser ensamblado de la siguiente manera:



```
Virtual Terminal
Transmision de datos infrarrojos.
Esta emision esta modulada a 38K Hz.
Con una velocidad de transmision de 2400 bps.

Transmision de datos infrarrojos.
Esta emision esta modulada a 38K Hz.
Con una velocidad de transmision de 2400 bps.

Transmision de datos infrarroj|
```



Sergio
Velásquez

Comunicación con memorias SD

- Las memorias de almacenamiento masivo SD, pueden ser manipuladas por los microcontroladores de alta gama como la familia 18F. Para leer o escribir sobre una memoria SD, se requiere trabajar con segmentos de memoria de 512 bytes. Para este fin es necesario contar con espacios de memoria RAM, de gran capacidad como los microcontroladores de la familia 18F. La comunicación de las memorias SD, requieren de un protocolo serial conocido como SPI, para este fin los microcontroladores cuentan con un módulo de comunicación serial de este tipo. La alimentación de voltaje de las memorias SD es de 3 voltios, para esto se requiere hacer un acople de voltajes con el microcontrolador por medio de un divisor de voltaje con resistencias. La apariencia física y la distribución de pines de este tipo de memorias son las siguientes:



Comunicación con memorias SD

- La capacidad de las memorias SD actuales es del orden de los Giga bytes, sin embargo la máxima capacidad que soporta la librería para estas memorias en MikroC PRO, es de 1G bytes. Para la manipulación de este tipo de memorias el compilador MikroC PRO, cuenta con una librería denominada: Mmc. La cual posee tres funciones para establecer de forma simple la comunicación y control de la memoria. La función: `unsigned char Mmc_Init( void );` esta función inicializa la comunicación con la memoria SD, y retorna 0 si la comunicación se estableció exitosamente, o retorna 1 si algún error se presenta. La función: `unsigned char Mmc_Read_Sector(unsigned long sector, char *dbuff);` esta permite leer un sector de 512 bytes, guardando la lectura en el apuntador dbuff, e iniciando en el sector definido por el parámetro sector. La función: `unsigned char Mmc_Write_Sector(unsigned long sector, char *dbuff);` esta función trabaja de forma similar a la función de lectura, en las dos funciones se retorna el valor 0 si la operación culmina exitosamente o retornan 1 si un fallo se presenta. Para la comunicación física se requieren 4 conexiones que son: una línea de reloj, dos líneas de datos, entrada y salida, y un selector de pastilla.

Comunicación con memorias SD

- Para contextualizar el funcionamiento de estos dispositivos se puede observar y analizar el siguiente código fuente:

```
// Pines de conexión para el selector de pastilla
sbit Mmc_Chip_Select      at RC2_bit;
sbit Mmc_Chip_Select_Direction at TRISC2_bit;

void main( void )
{
    //Declaración de variables.
    char SD;
    char Texto[50];
    long SEC=0;
    unsigned int Cont;
    unsigned short Menu=0;
    unsigned short NN;
    char Dato;
    unsigned short Bufer[512];
    //Inicio del puerto serial a 9600 bps.
    UART1_Init(9600);
    //Inicio del puerto SPI, para la memoria SD.
    SPI1_Init_Advanced( _SPI_MASTER_OSC_DIV64, _SPI_DATA_SAMPLE_MIDDLE,
        _SPI_CLK_IDLE_HIGH, _SPI_LOW_2_HIGH);
    //Inicio de la memoria.
```

Comunicación con memorias SD

```
if( Mmc_Init()==0 )
{
    //Confirmación exitosa de conexión.
    SD = 1;
    UART1_Write_Text("// La memoria fue iniciada exitosamente. //");
    UART1_Write(13);
    UART1_Write(10);
    UART1_Write(13);
    UART1_Write(10);
}
else
{
    //Confirmación fallida de conexión.
    SD = 0;
    UART1_Write_Text("ERROR no se puede iniciar la memoria !!!");
    UART1_Write(13);
    UART1_Write(10);
    UART1_Write(13);
    UART1_Write(10);
}
```

Comunicación con memorias SD

```
while(1) //Bucle infinito.
{
    if( SD ) //Se ejecuta el Menú si la conexión fue exitosa.
    {
        //Se despliega el Menú.
        UART1_Write_Text( "1. Grabar un Sector." );
        UART1_Write(13);
        UART1_Write(10);
        UART1_Write_Text( "2. Leer un Sector." );
        UART1_Write(13);
        UART1_Write(10);
        UART1_Write_Text( "Digite una opcion: " );
        Menu = 0;
        //Bucle para esperar la opción.

        while( Menu==0 )
        {
            if( UART1_Data_Ready() )
            {
                Dato = UART1_Read();
                UART1_Write(Dato);
                UART1_Write(' ');
                switch( Dato )
                {
                    case '1': Menu=1; break;
                    case '2': Menu=2; break;
                    default: break;
                }
            }
        }
    }
}
```

Comunicación con memorias SD

//Envío del enter y retroceso del carro.

```
UART1_Write(13);
```

```
UART1_Write(10);
```

//Captura del sector de lectura o escritura.

```
UART1_Write_Text( "Digite el numero del sector de memoria, y pulse ENTER: " );
```

```
SEC=-1;
```

```
NN=0;
```

```
Texto[0]=0;
```

//Bucle para digital el sector.

```
while( SEC!=-1 )
```

```
{
```

```
    if( UART1_Data_Ready() )
```

```
    {
```

```
        Dato = UART1_Read();
```

```
        UART1_Write(Dato);
```

```
        switch( Dato )
```

```
        {
```

```
            case '1':
```

```
            case '2':
```

```
            case '3':
```

```
            case '4':
```

```
            case '5':
```

```
            case '6':
```

```
            case '7':
```

```
            case '8':
```

```
            case '9':
```

```
            case '0': Texto[NN++]=Dato;
```

```
                    Texto[NN]=0;
```

```
                    break;
```

```
            case 13: SEC=atoi(Texto);
```

```
            default : break;
```

```
        }
```

```
    }
```

```
}
```

Comunicación con memorias SD

```
//Envío del enter y retroceso del carro.
UART1_Write(13);
UART1_Write(10);
//Se graban o se leen los sectores.
if( Menu==1 )
{
    UART1_Write_Text( "Se grabara el sector: " );
    LongToStr( SEC, Texto);
    UART1_Write_Text( Texto );
    //Envío del enter y retroceso del carro.
    UART1_Write(13);
    UART1_Write(10);
    //Se borra el Búfer.
    for( Cont=0; Cont<512; Cont++ )
        Bufer[Cont]=0;
    //Bucle para capturar los datos a grabar.
}
```



**Sergio
Velásquez**

Comunicación con memorias SD

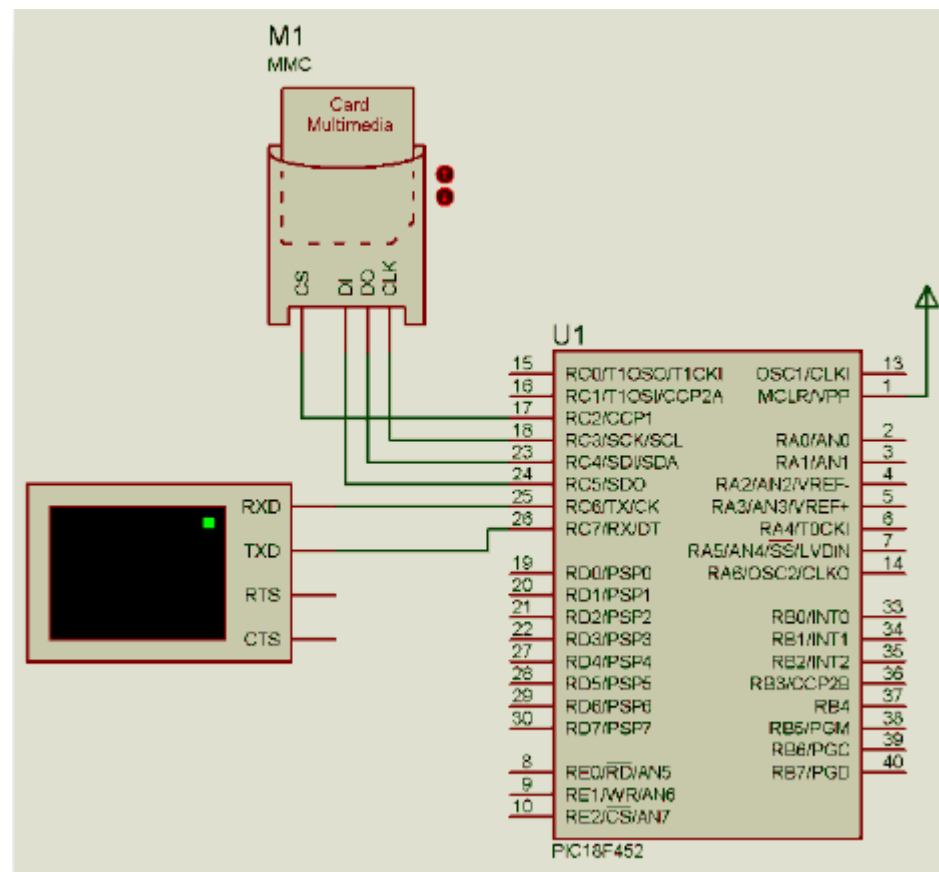
```
while( Menu )
{
    if( UART1_Data_Ready() )
    {
        Dato = UART1_Read();
        UART1_Write(Dato);
        switch( Dato )
        {
            case 13: Menu=0; break;
            default : Bufer[NN++]=Dato;
                    break;
        }
    }
}
//Se graban los 512 datos del sector en la memoria.
Mmc_Write_Sector( SEC, Bufer);
//Envío del enter y retroceso del carro.
UART1_Write(13);
UART1_Write(10);
UART1_Write_Text( "Fin de la grabacion." );
}
```

Comunicación con memorias SD

```
else
{
    UART1_Write_Text( "Se leera el sector: " );
    LongToStr( SEC, Texto);
    UART1_Write_Text( Texto );
    //Envío del enter y retroceso del carro.
    UART1_Write(13);
    UART1_Write(10);
    //Se leen 512 datos del sector.
    Mmc_Read_Sector( SEC, Bufer);
    //Bucle para imprimir los datos leídos de la memoria.
    for( Cont=0; Cont<512; Cont++ )
        UART1_Write( Bufer[Cont] );
}
//Envío del enter y retroceso del carro.
UART1_Write(13);
UART1_Write(10);
//Envío del enter y retroceso del carro.
UART1_Write(13);
UART1_Write(10);
}
```

Comunicación con memorias SD

- Como medida adicional se debe inicializar el puerto SPI, por medio de la función: *SPI1_Init_Advanced*. Para comprobar el funcionamiento de este programa se debe construir en ISIS, un circuito con los dispositivos: 18F452, MMC, y VIRTUAL TERMINAL, como se puede apreciar en la siguiente gráfica:



Comunicación con memorias SD

- Para fines de simulación las conexiones con la memoria se hacen directamente como se puede ver en el anterior circuito. Sin embargo se debe tener presente que para fines prácticos la memoria trabaja con 3 voltios, y se debe acoplar con un arreglo de resistencias como el siguiente:

